Requests, NumPy & Pandas, ...

План

- "Магические команды"
- NumPy
- Pandas
- Regex*

In [1]:
```
%load_ext nb_black
```

Magic commands

Специальные команды, которые могут помочь с запуском и анализом данных в ваших рабочих тетрадках.

In [2]:
```
%lsmagic
```

Out[2]:
```
Available line magics:
%alias  %alias_magic  %autoawait  %autocall  %automagic  %autosave
%bookmark  %cat  %cd  %clear  %colors  %conda  %config  %connect_i
nfo  %cp  %debug  %dhist  %dirs  %doctest_mode  %ed  %edit  %env
%gui  %hist  %history  %killbgscripts  %ldir  %less  %lf  %lk  %ll
%load  %load_ext  %loadpy  %logoff  %logon  %logstart  %logstate
%logstop  %ls  %lsmagic  %lx  %macro  %magic  %man  %matplotlib  %
mkdir  %more  %mv  %notebook  %page  %pastebin  %pdb  %pdef  %pdoc
%pfile  %pinfo  %pinfo2  %pip  %popd  %pprint  %precision  %prun
%psearch  %psource  %pushd  %pwd  %pycat  %pylab  %qtconsole  %qui
ckref  %recall  %rehashx  %reload_ext  %rep  %rerun  %reset  %rese
t_selective  %rm  %rmdir  %run  %save  %sc  %set_env  %store  %sx
%system  %tb  %time  %timeit  %unalias  %unload_ext  %who  %who_ls
%whos  %xdel  %xmode

Available cell magics:
%%!  %%HTML  %%SVG  %%bash  %%capture  %%debug  %%file  %%html  %%
javascript  %%js  %%latex  %%markdown  %%perl  %%prun  %%pypy  %%p
ython  %%python2  %%python3  %%ruby  %%script  %%sh  %%svg  %%sx
%%system  %%time  %%timeit  %%writefile

Automagic is ON, % prefix IS NOT needed for line magics.
```

In [3]:
```
%run main.py
```

```
Run a script
```

In [4]:
```
a, b, c, d = 1, 2, 3, 4

%who
%who str
%who int
```

```
a	b	c	d
No variables match your requested type.
a	b	c	d
```

In [5]:
```
%pinfo a
```

In [6]:
```
%env NEW_VAR=123
%env NEW_VAR
```

```
env: NEW_VAR=123
```

Out[6]:
```
'123'
```

In [7]:
```
# %load main.py
print("Run a script")
```

```
Run a script
```

NumPy

https://numpy.org (https://numpy.org)

Библиотека добавляющая удобную поддержку работы с многомерными массивами и матрицами. В ней присутствует большая библиотека высокоуровневых математических функций для операций с ними.

```
# https://numpy.org/install/
!pip3 install numpy
```

In [8]:
```
import numpy as np  # подключение библиотеки
```

Тонкости NumPy объектов

In [9]:
```
arr = np.arange(1000000)  # range like только в numpy
lst = list(range(1000000))

print(arr)
```

```
[     0      1      2 ... 999997 999998 999999]
```

In [10]:
```
%time for _ in range(10): arr_2 = arr * 2
%time for _ in range(10): lst_2 = [x * 2 for x in lst]
```

```
CPU times: user 20.9 ms, sys: 6.45 ms, total: 27.4 ms
Wall time: 25.8 ms
CPU times: user 638 ms, sys: 190 ms, total: 828 ms
Wall time: 836 ms
```

In [11]:
```
arr = np.arange(0, 20, 3)  # работает как и range
arr
```

Out[11]:
```
array([ 0,  3,  6,  9, 12, 15, 18])
```

In [12]:
```
# хотим из вектора сделать матрицу -> изменяем размерность
arr = np.arange(20)
matrix = arr.reshape(4, 5)
matrix
```

Out[12]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
```

In [13]:
```
matrix.ravel()  # обратно, из матрицы делаем вектор
matrix.flatten()
```

Out[13]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14,
15, 16,
       17, 18, 19])
```

In [14]:
```
np.arange(32).reshape(4, 2, 4)  # 3-мерная матрица
```

Out[14]:
```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7]],

       [[ 8,  9, 10, 11],
        [12, 13, 14, 15]],

       [[16, 17, 18, 19],
        [20, 21, 22, 23]],

       [[24, 25, 26, 27],
        [28, 29, 30, 31]]])
```

In [15]:
```
print(matrix.shape)  # размерность матрицы
print(matrix.ndim)  # размерность пространства
print(matrix.itemsize)  # сколько занимает в памяти один элемен
print(matrix.size)  # количество элементов
```

```
(4, 5)
2
8
20
```

In [16]:
```
arr1d = np.array([1.0, 2.7, 3.1])  # вещественные числа
arr1d.dtype, arr1d.itemsize
```

Out[16]:
```
(dtype('float64'), 8)
```

In [17]:
```
print(
    f"32: {np.float32(0.1) + np.float32(0.2) == np.float32(0.3)}"
    f"64: {np.float64(0.1) + np.float64(0.2) == np.float64(0.3)}"
)
```

```
32: True 64: False
```

In [18]:
```
arr2d = np.array([[1.0, 2.7, 3.1], (8.3, 3.14, 2.7)])
arr2d
```

Out[18]:
```
array([[1.  , 2.7 , 3.1 ],
       [8.3 , 3.14, 2.7 ]])
```

In [19]:
```
floats = np.linspace(0, 5, 21)  # range но с вещественным шагом
ints = np.arange(21)

print(floats)
print()
print(ints)
```

```
[0.   0.25 0.5  0.75 1.   1.25 1.5  1.75 2.   2.25 2.5  2.75 3.
3.25
 3.5  3.75 4.   4.25 4.5  4.75 5.  ]

[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
```

In [20]:
```
print(ints + floats)
print(ints * 2)
print(ints > 10)
print(ints @ floats)  # векторное умножение
```

```
[ 0.    1.25  2.5   3.75  5.    6.25  7.5   8.75 10.   11.25 12.5
13.75
 15.   16.25 17.5  18.75 20.   21.25 22.5  23.75 25.  ]
[ 0  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40]
[False False False False False False False False False False False
True
  True  True  True  True  True  True  True  True  True]
717.5
```

In [21]:
```
ints[ints > 10]  # мы можем сразу посмотреть, где True
```

Out[21]:
```
array([11, 12, 13, 14, 15, 16, 17, 18, 19, 20])
```

In [22]:
```
arr = ints * floats
print(arr)
print(np.where(arr < 25))  # возвращает индексы
print(
    np.where(arr < 25, arr, -1)
)  # возвращает элементы arr, если выполняется условия, иначе -
print(
    np.where(arr < 25, 1, -1)
)  # возвращает элементы arr, если выполняется условия, иначе -
```

```
[  0.     0.25   1.     2.25   4.     6.25   9.    12.25  16.    2
0.25
  25.    30.25  36.    42.25  49.    56.25  64.    72.25  81.    9
0.25
 100.  ]
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),)
[ 0.    0.25  1.    2.25  4.    6.25  9.   12.25 16.   20.25 -1.
-1.
 -1.   -1.   -1.   -1.   -1.   -1.   -1.   -1.   -1.   ]
[ 1  1  1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
```

In [23]:
```
np.zeros((3, 3))  # создаем матрицу из вещественных нулей
```

Out[23]:
```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

In [24]:
```
np.ones((3, 3))  # создаем матрицу из вещественных единиц
```

Out[24]:
```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

In [25]:
```
# Генератор псевдослучайных чисел
rand_gen = np.random.default_rng(1)  # сам генератор

M1 = rand_gen.random((3, 3))
M2 = rand_gen.random((1, 3))
M1, M2
```

Out[25]:
```
(array([[0.51182162, 0.9504637 , 0.14415961],
        [0.94864945, 0.31183145, 0.42332645],
        [0.82770259, 0.40919914, 0.54959369]]),
 array([[0.02755911, 0.75351311, 0.53814331]]))
```

In [26]:
```
M1.T  # транспонирование матрицы
```

Out[26]:
```
array([[0.51182162, 0.94864945, 0.82770259],
       [0.9504637 , 0.31183145, 0.40919914],
       [0.14415961, 0.42332645, 0.54959369]])
```

In [27]:
```
M1 * M2  # поэлементное
```

Out[27]:
```
array([[0.01410535, 0.71618685, 0.07757853],
       [0.02614394, 0.23496909, 0.2278103 ],
       [0.02281075, 0.30833691, 0.29576017]])
```

In [28]:
```
M1 @ M2.T  # матричное
M1.dot(M2.T)
```

Out[28]:
```
array([[0.80787074],
       [0.48892332],
       [0.62690783]])
```

Доступ к элементам

In [29]:
```
arr = np.arange(100)
matrix = np.arange(100).reshape(10, 10)
matrix
```

Out[29]:
```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
       [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
       [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

In [30]:
```
arr[0] = 1000
arr[10:20] = -1  # можем сразу изменить срез
arr[::-1]
```

Out[30]:
```
array([  99,   98,   97,   96,   95,   94,   93,   92,   91,   90,
89,
         88,   87,   86,   85,   84,   83,   82,   81,   80,   79,
78,
         77,   76,   75,   74,   73,   72,   71,   70,   69,   68,
67,
         66,   65,   64,   63,   62,   61,   60,   59,   58,   57,
56,
         55,   54,   53,   52,   51,   50,   49,   48,   47,   46,
45,
         44,   43,   42,   41,   40,   39,   38,   37,   36,   35,
34,
         33,   32,   31,   30,   29,   28,   27,   26,   25,   24,
23,
         22,   21,   20,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
-1,
         -1,   -1,    9,    8,    7,    6,    5,    4,    3,    2,
1,
       1000])
```

In [31]:
```
matrix[:3]  # строки
matrix[1:3, :5]  # строки, столбцы
```

Out[31]:
```
array([[10, 11, 12, 13, 14],
       [20, 21, 22, 23, 24]])
```

In [32]:
```
vec1 = np.arange(0, 10, 2)
vec2 = np.arange(5)
print(vec1)
print(vec2)
print()

h = np.hstack([vec1, vec2])
v = np.vstack([vec1, vec2])
print(h)
print(v)
```

```
[0 2 4 6 8]
[0 1 2 3 4]

[0 2 4 6 8 0 1 2 3 4]
[[0 2 4 6 8]
 [0 1 2 3 4]]
```

Pandas

https://pandas.pydata.org (https://pandas.pydata.org)

Одна из "главных" библиотек для анализа данных, позволяет работать с данными в привычном для нас табличном виде.

Главные структуры данных библиотеки:

- DataFrame -- таблица
- Series -- столбец / вектор

```
# https://pandas.pydata.org/getting_started.html
!pip install pandas
```

In [33]:
```
import pandas as pd
```

In [34]:
```
pd.Series([1, 2, 3])
```

Out[34]:
```
0    1
1    2
2    3
dtype: int64
```

In [35]:
```
pd.DataFrame(np.arange(27).reshape(9, 3))
```

Out[35]:

	0	1	2
0	0	1	2
1	3	4	5
2	6	7	8
3	9	10	11
4	12	13	14
5	15	16	17
6	18	19	20
7	21	22	23
8	24	25	26

In [36]:
```
pd.DataFrame(
    np.arange(27).reshape(9, 3),
    columns=["First", "Second", "Third"],
    index=range(9, 0, -1),
)
```

Out[36]:

	First	Second	Third
9	0	1	2
8	3	4	5
7	6	7	8
6	9	10	11
5	12	13	14
4	15	16	17
3	18	19	20
2	21	22	23
1	24	25	26

Будем работать с реальными табличными данными

https://www.kaggle.com/lava18/google-play-store-apps?select=googleplaystore.csv (https://www.kaggle.com/lava18/google-play-store-apps?select=googleplaystore.csv)

In [37]:
```
with open("googleplaystore.csv", "r") as file:
    for i, line in enumerate(file):
        print(line)
        if i > 4:
            break
```

```
App,Category,Rating,Reviews,Size,Installs,Type,Price,Content Rating,Genres,Last Updated,Current Ver,Android Ver

Photo Editor & Candy Camera & Grid & ScrapBook,ART_AND_DESIGN,4.1,159,19M,"10,000+",Free,0,Everyone,Art & Design,"January 7, 2018",1.0.0,4.0.3 and up

Coloring book moana,ART_AND_DESIGN,3.9,967,14M,"500,000+",Free,0,Everyone,Art & Design;Pretend Play,"January 15, 2018",2.0.0,4.0.3 and up

"U Launcher Lite – FREE Live Cool Themes, Hide Apps",ART_AND_DESIGN,4.7,87510,8.7M,"5,000,000+",Free,0,Everyone,Art & Design,"August 1, 2018",1.2.4,4.0.3 and up

Sketch - Draw & Paint,ART_AND_DESIGN,4.5,215644,25M,"50,000,000+",Free,0,Teen,Art & Design,"June 8, 2018",Varies with device,4.2 and up

Pixel Draw - Number Art Coloring Book,ART_AND_DESIGN,4.3,967,2.8M,"100,000+",Free,0,Everyone,Art & Design;Creativity,"June 20, 2018",1.1,4.4 and up
```

In [38]:
```
df = pd.read_csv("googleplaystore.csv")
```

In [39]:
```
type(df)
```

Out[39]:
```
pandas.core.frame.DataFrame
```

In [40]:
```
df.head()  # возвращает первые n=5 строк
```

Out[40]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Conten Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone

In [41]:
```
df.tail(2)  # конец
```

Out[41]:

	App	Category	Rating	Reviews	Size	Installs	Type	P
10839	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5	114	Varies with device	1,000+	Free	
10840	iHoroscope - 2018 Daily Horoscope & Astrology	LIFESTYLE	4.5	398307	19M	10,000,000+	Free	

In [42]:
```
1 df.describe(include="all")  # статистика по нашей таблице
```

Out[42]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Cont Rat
count	10841	10841	9367.000000	10841	10841	10841	10840	10841	10
unique	9660	34	NaN	6002	462	22	3	93	
top	ROBLOX	FAMILY	NaN	0	Varies with device	1,000,000+	Free	0	Every
freq	9	1972	NaN	596	1695	1579	10039	10040	8
mean	NaN	NaN	4.193338	NaN	NaN	NaN	NaN	NaN	N
std	NaN	NaN	0.537431	NaN	NaN	NaN	NaN	NaN	N
min	NaN	NaN	1.000000	NaN	NaN	NaN	NaN	NaN	N
25%	NaN	NaN	4.000000	NaN	NaN	NaN	NaN	NaN	N
50%	NaN	NaN	4.300000	NaN	NaN	NaN	NaN	NaN	N
75%	NaN	NaN	4.500000	NaN	NaN	NaN	NaN	NaN	N
max	NaN	NaN	19.000000	NaN	NaN	NaN	NaN	NaN	N

ссылка: https://birdyx.ru/blog/show/data-analysis-1-part (https://birdyx.ru/blog/show/data-analysis-1-part)

In [43]:
```
df.shape  # размерность таблицы
```

Out[43]:
```
(10841, 13)
```

In [44]:
```
df["Category"]  # доступ к колонке
```

Out[44]:
```
0              ART_AND_DESIGN
1              ART_AND_DESIGN
2              ART_AND_DESIGN
3              ART_AND_DESIGN
4              ART_AND_DESIGN
                  ...
10836                  FAMILY
10837                  FAMILY
10838                 MEDICAL
10839    BOOKS_AND_REFERENCE
10840               LIFESTYLE
Name: Category, Length: 10841, dtype: object
```

In [45]:
```
type(df.Category)
```

Out[45]:
```
pandas.core.series.Series
```

In [46]:
```
df["Content Rating"]  # df.Content Rating
```

Out[46]:
```
0          Everyone
1          Everyone
2          Everyone
3              Teen
4          Everyone
             ...
10836      Everyone
10837      Everyone
10838      Everyone
10839    Mature 17+
10840      Everyone
Name: Content Rating, Length: 10841, dtype: object
```

In [47]:
```
df[["App", "Category", "Rating"]]  # несколько колонок
```

Out[47]:

	App	Category	Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1
1	Coloring book moana	ART_AND_DESIGN	3.9
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3
...	...	...	...
10836	Sya9a Maroc - FR	FAMILY	4.5
10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0
10838	Parkinson Exercices FR	MEDICAL	NaN
10839	The SCP Foundation DB fr nn5n	BOOKS_AND_REFERENCE	4.5
10840	iHoroscope - 2018 Daily Horoscope & Astrology	LIFESTYLE	4.5

10841 rows × 3 columns

In [48]:
```
df[5:10]  # срез по строкам
```

Out[48]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
5	Paper flowers instructions	ART_AND_DESIGN	4.4	167	5.6M	50,000+	Free	0	Everyone
6	Smoke Effect Photo Maker - Smoke Editor	ART_AND_DESIGN	3.8	178	19M	50,000+	Free	0	Everyone
7	Infinite Painter	ART_AND_DESIGN	4.1	36815	29M	1,000,000+	Free	0	Everyone
8	Garden Coloring Book	ART_AND_DESIGN	4.4	13791	33M	1,000,000+	Free	0	Everyone
9	Kids Paint Free - Drawing Fun	ART_AND_DESIGN	4.7	121	3.1M	10,000+	Free	0	Everyone

In [49]:
```
df.Rating == 4.7  # можем сравнить со всеми значениями в колонк
```

Out[49]:
```
0        False
1        False
2         True
3        False
4        False
         ...
10836    False
10837    False
10838    False
10839    False
10840    False
Name: Rating, Length: 10841, dtype: bool
```

In [50]:
```
df[df.Rating > 4.7]  # вернутся только те записи, которые True
```

Out[50]:

	App	Category	Rating	Reviews	Size	Installs	Type	P
25	Harley Quinn wallpapers HD	ART_AND_DESIGN	4.8	192	6.0M	10,000+	Free	
55	Tickets SDA 2018 and Exam from the State Traff...	AUTO_AND_VEHICLES	4.9	10479	33M	100,000+	Free	
61	CDL Practice Test 2018 Edition	AUTO_AND_VEHICLES	4.9	7774	17M	100,000+	Free	
64	DMV Permit Practice Test 2018 Edition	AUTO_AND_VEHICLES	4.9	6090	27M	100,000+	Free	
70	Fines of the State Traffic Safety Inspectorate...	AUTO_AND_VEHICLES	4.8	116986	35M	5,000,000+	Free	
...	...	...	...	...	...	...	...	
10801	Fr Ignacio Outreach	FAMILY	4.9	52	19M	1,000+	Free	
10810	Fr Lupupa Sermons	BUSINESS	4.8	19	21M	100+	Free	
10820	Fr. Daoud Lamei	FAMILY	5.0	22	8.6M	1,000+	Free	
10833	Chemin (fr)	BOOKS_AND_REFERENCE	4.8	44	619k	1,000+	Free	
10837	Fr. Mike Schmitz Audio Teachings	FAMILY	5.0	4	3.6M	100+	Free	

596 rows × 13 columns

In [51]:
```
df_rating_4_7 = df[df.Rating == 4.7]  # создадим новый df для R
```

In [52]:
```
df_rating_4_7.head()
```

Out[52]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Conte Rati
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyc
9	Kids Paint Free - Drawing Fun	ART_AND_DESIGN	4.7	121	3.1M	10,000+	Free	0	Everyc
16	Photo Designer - Write your name with shapes	ART_AND_DESIGN	4.7	3632	5.5M	500,000+	Free	0	Everyc
22	Superheroes Wallpapers | 4K Backgrounds	ART_AND_DESIGN	4.7	7699	4.2M	500,000+	Free	0	Everyc 1
24	HD Mickey Minnie Wallpapers	ART_AND_DESIGN	4.7	118	23M	50,000+	Free	0	Everyc

In [53]:
```
df_rating_4_7.iloc[0:5, 1:5]  # получить по порядку
```

Out[53]:

	Category	Rating	Reviews	Size
2	ART_AND_DESIGN	4.7	87510	8.7M
9	ART_AND_DESIGN	4.7	121	3.1M
16	ART_AND_DESIGN	4.7	3632	5.5M
22	ART_AND_DESIGN	4.7	7699	4.2M
24	ART_AND_DESIGN	4.7	118	23M

In [54]:
```
df_rating_4_7.loc[0:5]  # получить по значению
```

Out[54]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone

In [55]:
```
df_rating_4_7.loc[0:10, ["App", "Reviews"]]  # по значению и ин
```

Out[55]:

	App	Reviews
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	87510
9	Kids Paint Free - Drawing Fun	121

In [56]:
```
df_rating_4_7.index, df_rating_4_7.values
```

Out[56]:
```
(Int64Index([    2,     9,    16,    22,    24,    26,    34,    3
6,    38,
                45,
             ...
             10459, 10484, 10571, 10576, 10593, 10604, 10628, 1079
6, 10799,
             10809],
            dtype='int64', length=499),
 array([['U Launcher Lite – FREE Live Cool Themes, Hide Apps',
         'ART_AND_DESIGN', 4.7, ..., 'August 1, 2018', '1.2.4',
         '4.0.3 and up'],
        ['Kids Paint Free - Drawing Fun', 'ART_AND_DESIGN', 4.7, .
..,
         'July 3, 2018', '2.8', '4.0.3 and up'],
        ['Photo Designer - Write your name with shapes', 'ART_AND_
DESIGN',
         4.7, ..., 'July 31, 2018', '3.1', '4.1 and up'],
        ...,
        ['Inf VPN - Global Proxy & Unlimited Free WIFI VPN', 'TOOL
S', 4.7,
         ..., 'July 26, 2018', '1.9.734', '4.1 and up'],
        ['Fr Daoud Lamei', 'SOCIAL', 4.7, ..., 'May 20, 2018', '1.
72',
         '4.0.3 and up'],
        ['Castle Clash: RPG War and Strategy FR', 'FAMILY', 4.7, .
..,
         'July 18, 2018', '1.4.2', '4.1 and up']], dtype=object))
```

In [57]:
```
df_rating_4_7.iloc[0:5, [0, 3]]  # по порядку и индекса и колон
```

Out[57]:

	App	Reviews
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	87510
9	Kids Paint Free - Drawing Fun	121
16	Photo Designer - Write your name with shapes	3632
22	Superheroes Wallpapers | 4K Backgrounds	7699
24	HD Mickey Minnie Wallpapers	118

In [58]:
```
df_rating_4_7.at[2, "App"], df_rating_4_7.iat[0, 0]  # получить
```

Out[58]:
```
('U Launcher Lite – FREE Live Cool Themes, Hide Apps',
 'U Launcher Lite – FREE Live Cool Themes, Hide Apps')
```

In [59]:
```
df_temp = df.copy()  # df[:5]
df_temp.iloc[0, 0] = "NEW VALUE 2"
df.head()  # -> df.copy()
```

Out[59]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Conten Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone

Чуть проанализирум данные

In [60]:
```
1 df.head()
```

Out[60]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Conten Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone

Каких категорий больше в play market?

In [61]:
```
1 df.Category
```

Out[61]:
```
0              ART_AND_DESIGN
1              ART_AND_DESIGN
2              ART_AND_DESIGN
3              ART_AND_DESIGN
4              ART_AND_DESIGN
                  ...
10836                  FAMILY
10837                  FAMILY
10838                 MEDICAL
10839    BOOKS_AND_REFERENCE
10840               LIFESTYLE
Name: Category, Length: 10841, dtype: object
```

In [62]:
```
1 df.Category.value_counts()  # метод для Series
```

Out[62]:
```
FAMILY                 1972
GAME                   1144
TOOLS                   843
MEDICAL                 463
BUSINESS                460
PRODUCTIVITY            424
PERSONALIZATION         392
COMMUNICATION           387
SPORTS                  384
LIFESTYLE               382
FINANCE                 366
HEALTH_AND_FITNESS      341
PHOTOGRAPHY             335
SOCIAL                  295
NEWS_AND_MAGAZINES      283
SHOPPING                260
TRAVEL_AND_LOCAL        258
DATING                  234
BOOKS_AND_REFERENCE     231
VIDEO_PLAYERS           175
EDUCATION               156
ENTERTAINMENT           149
MAPS_AND_NAVIGATION     137
FOOD_AND_DRINK          127
HOUSE_AND_HOME           88
AUTO_AND_VEHICLES        85
LIBRARIES_AND_DEMO       85
WEATHER                  82
ART_AND_DESIGN           65
EVENTS                   64
COMICS                   60
PARENTING                60
BEAUTY                   53
1.9                       1
Name: Category, dtype: int64
```

In [63]:
```
df.Category.value_counts().index, df.Category.value_counts().va
```

Out[63]:
```
(Index(['FAMILY', 'GAME', 'TOOLS', 'MEDICAL', 'BUSINESS', 'PRODUCT
IVITY',
       'PERSONALIZATION', 'COMMUNICATION', 'SPORTS', 'LIFESTYLE',
'FINANCE',
       'HEALTH_AND_FITNESS', 'PHOTOGRAPHY', 'SOCIAL', 'NEWS_AND_M
AGAZINES',
       'SHOPPING', 'TRAVEL_AND_LOCAL', 'DATING', 'BOOKS_AND_REFER
ENCE',
       'VIDEO_PLAYERS', 'EDUCATION', 'ENTERTAINMENT', 'MAPS_AND_N
AVIGATION',
       'FOOD_AND_DRINK', 'HOUSE_AND_HOME', 'AUTO_AND_VEHICLES',
       'LIBRARIES_AND_DEMO', 'WEATHER', 'ART_AND_DESIGN', 'EVENTS
', 'COMICS',
       'PARENTING', 'BEAUTY', '1.9'],
      dtype='object'),
 array([1972, 1144,  843,  463,  460,  424,  392,  387,  384,  382
,  366,
         341,  335,  295,  283,  260,  258,  234,  231,  175,  156
,  149,
         137,  127,   88,   85,   85,   82,   65,   64,   60,   60
,   53,
           1]))
```

Средняя длина названия приложения?

In [64]:
```
df.App
```

Out[64]:
```
0          Photo Editor & Candy Camera & Grid & ScrapBook
1                                      Coloring book moana
2       U Launcher Lite – FREE Live Cool Themes, Hide ...
3                                    Sketch - Draw & Paint
4                  Pixel Draw - Number Art Coloring Book
                              ...
10836                                    Sya9a Maroc - FR
10837                  Fr. Mike Schmitz Audio Teachings
10838                           Parkinson Exercices FR
10839                     The SCP Foundation DB fr nn5n
10840       iHoroscope - 2018 Daily Horoscope & Astrology
Name: App, Length: 10841, dtype: object
```

In [65]:
```
df.App.apply(len)
```

Out[65]:
```
0        46
1        19
2        50
3        21
4        37
         ..
10836    16
10837    32
10838    22
10839    29
10840    45
Name: App, Length: 10841, dtype: int64
```

In [66]:
```
df.App.apply(lambda x: len(x.split()))
```

Out[66]:
```
0         9
1         3
2        10
3         5
4         7
         ..
10836     4
10837     5
10838     3
10839     6
10840     7
Name: App, Length: 10841, dtype: int64
```

In [67]:
```
print(
    df.App.apply(lambda x: len(x.split())).mean(),  # средняя
    df.App.apply(lambda x: len(x.split())).median(),  # медиана
    df.App.apply(lambda x: len(x.split())).max(),  # максимальн
    df.App.apply(lambda x: len(x.split())).min(),  # минимальна
    df.App.apply(lambda x: len(x.split())).quantile(0.25),  # к
    df.App.apply(lambda x: len(x.split())).quantile(0.99),  # к
)
```

```
3.9280509178120098 3.0 26 1 2.0 10.0
```

Сколько всего скачиваний?

In [68]:
```
df.Installs.head()
```

Out[68]:
```
0         10,000+
1        500,000+
2      5,000,000+
3     50,000,000+
4        100,000+
Name: Installs, dtype: object
```

In [69]:
```
# df.Installs.sum(), sum(df.Installs)
```

In [70]:
```
df.Installs.dtype  # -> object, нам нужнен int
```

Out[70]:
```
dtype('O')
```

In [71]:
```
def installsToInt(install):
    install = "".join((i for i in install if i.isnumeric()))
    return int(install) if install else 0


df.Installs = df.Installs.apply(installsToInt)
```

In [72]:
```
df.Installs.head()
```

Out[72]:
```
0       10000
1      500000
2     5000000
3    50000000
4      100000
Name: Installs, dtype: int64
```

In [73]:
```
df.Installs.sum(), sum(df.Installs)
```

Out[73]:
```
(167633433487, 167633433487)
```

Какая в среднем цена на категорию и какая средняя оценка в этой категории

In [74]:
```
1 df.head()
```

Out[74]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10000	Free	0	Everyone
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500000	Free	0	Everyone
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5000000	Free	0	Everyone
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50000000	Free	0	Teen
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100000	Free	0	Everyone

In [75]:
```
1 df.Price.value_counts()
```

Out[75]:
```
0          10040
$0.99        148
$2.99        129
$1.99         73
$4.99         72
           ...
$3.28          1
$46.99         1
$4.77          1
$400.00        1
$299.99        1
Name: Price, Length: 93, dtype: int64
```

In [76]:
```
1 df.Price.apply(lambda x: np.float64(0.0) if x == "0" else np.fl
```

```
---------------------------------------------------------------------
---------
ValueError                                Traceback (most recent c
all last)
<ipython-input-76-37b1a7de38df> in <module>
----> 1 df.Price.apply(lambda x: np.float64(0.0) if x == "0" else
np.float64(x[1:]))

~/anaconda3/lib/python3.8/site-packages/pandas/core/series.py in a
pply(self, func, convert_dtype, args, **kwds)
   4136             else:
   4137                 values = self.astype(object)._values
-> 4138                 mapped = lib.map_infer(values, f, convert=
convert_dtype)
   4139
   4140         if len(mapped) and isinstance(mapped[0], Series):

pandas/_libs/lib.pyx in pandas._libs.lib.map_infer()

<ipython-input-76-37b1a7de38df> in <lambda>(x)
----> 1 df.Price.apply(lambda x: np.float64(0.0) if x == "0" else
np.float64(x[1:]))

ValueError: could not convert string to float: 'veryone'
```

In [77]:
```
1 df.Price.str.contains("veryone")
```

Out[77]:
```
0        False
1        False
2        False
3        False
4        False
         ...
10836    False
10837    False
10838    False
10839    False
10840    False
Name: Price, Length: 10841, dtype: bool
```

In [78]:
```
1 df.Price[df.Price.str.contains("veryone")]
```

Out[78]:
```
10472    Everyone
Name: Price, dtype: object
```

In [79]:

```
df.Price = df.Price.apply(
    lambda x: np.float64(0.0)
    if x == "0"
    else None
    if x == "Everyone"
    else np.float64(x[1:])
)
```

In [80]:

```
target_df = df[["Category", "Price", "Rating"]]
target_df
```

Out[80]:

	Category	Price	Rating
0	ART_AND_DESIGN	0.0	4.1
1	ART_AND_DESIGN	0.0	3.9
2	ART_AND_DESIGN	0.0	4.7
3	ART_AND_DESIGN	0.0	4.5
4	ART_AND_DESIGN	0.0	4.3
...	...	...	...
10836	FAMILY	0.0	4.5
10837	FAMILY	0.0	5.0
10838	MEDICAL	0.0	NaN
10839	BOOKS_AND_REFERENCE	0.0	4.5
10840	LIFESTYLE	0.0	4.5

10841 rows × 3 columns

In [81]:
```
1 target_df.fillna(0)  # заполнить пропуски 0
```

Out[81]:

	Category	Price	Rating
0	ART_AND_DESIGN	0.0	4.1
1	ART_AND_DESIGN	0.0	3.9
2	ART_AND_DESIGN	0.0	4.7
3	ART_AND_DESIGN	0.0	4.5
4	ART_AND_DESIGN	0.0	4.3
...	...	...	...
10836	FAMILY	0.0	4.5
10837	FAMILY	0.0	5.0
10838	MEDICAL	0.0	0.0
10839	BOOKS_AND_REFERENCE	0.0	4.5
10840	LIFESTYLE	0.0	4.5

10841 rows × 3 columns

In [82]:
```
1 target_df.Rating.fillna(target_df.Rating.mean())
```

Out[82]:
```
0        4.100000
1        3.900000
2        4.700000
3        4.500000
4        4.300000
           ...
10836    4.500000
10837    5.000000
10838    4.193338
10839    4.500000
10840    4.500000
Name: Rating, Length: 10841, dtype: float64
```

In [83]:
```
target_df.dropna()  # убрать все строчки с пропусками
```

Out[83]:

	Category	Price	Rating
0	ART_AND_DESIGN	0.0	4.1
1	ART_AND_DESIGN	0.0	3.9
2	ART_AND_DESIGN	0.0	4.7
3	ART_AND_DESIGN	0.0	4.5
4	ART_AND_DESIGN	0.0	4.3
...	...	...	...
10834	FAMILY	0.0	4.0
10836	FAMILY	0.0	4.5
10837	FAMILY	0.0	5.0
10839	BOOKS_AND_REFERENCE	0.0	4.5
10840	LIFESTYLE	0.0	4.5

9366 rows × 3 columns

In [84]:
```
target_df.dropna(inplace=True)  # не вернуть новую таблицу, а с
```

```
<ipython-input-84-1cd5ad8af614>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  target_df.dropna(inplace=True)  # не вернуть новую таблицу, а сразу заменить в target_df
```

In [85]:
```
target_df.shape
```

Out[85]: (9366, 3)

In [86]:
```
target_df.head()
```

Out[86]:

	Category	Price	Rating
0	ART_AND_DESIGN	0.0	4.1
1	ART_AND_DESIGN	0.0	3.9
2	ART_AND_DESIGN	0.0	4.7
3	ART_AND_DESIGN	0.0	4.5
4	ART_AND_DESIGN	0.0	4.3

In [87]:
```
target_df["Rating2.0"] = target_df.Rating.apply(lambda x: -x *
```

```
<ipython-input-87-a2197c69cebb>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  target_df["Rating2.0"] = target_df.Rating.apply(lambda x: -x * 2
)
```

In [88]:
```
target_df.head()
```

Out[88]:

	Category	Price	Rating	Rating2.0
0	ART_AND_DESIGN	0.0	4.1	-8.2
1	ART_AND_DESIGN	0.0	3.9	-7.8
2	ART_AND_DESIGN	0.0	4.7	-9.4
3	ART_AND_DESIGN	0.0	4.5	-9.0
4	ART_AND_DESIGN	0.0	4.3	-8.6

In [89]:
```
target_df.corr()
```

Out[89]:

	Price	Rating	Rating2.0
Price	1.000000	-0.021903	0.021903
Rating	-0.021903	1.000000	-1.000000
Rating2.0	0.021903	-1.000000	1.000000

In [90]:
```
target_df.drop([0, 1, 2], axis=0)  # убрать строки с 0, 1, 2 ин
```

Out[90]:

	Category	Price	Rating	Rating2.0
3	ART_AND_DESIGN	0.0	4.5	-9.0
4	ART_AND_DESIGN	0.0	4.3	-8.6
5	ART_AND_DESIGN	0.0	4.4	-8.8
6	ART_AND_DESIGN	0.0	3.8	-7.6
7	ART_AND_DESIGN	0.0	4.1	-8.2
...	...	...	...	...
10834	FAMILY	0.0	4.0	-8.0
10836	FAMILY	0.0	4.5	-9.0
10837	FAMILY	0.0	5.0	-10.0
10839	BOOKS_AND_REFERENCE	0.0	4.5	-9.0
10840	LIFESTYLE	0.0	4.5	-9.0

9363 rows × 4 columns

In [91]:
```
target_df.drop("Category", axis=1)  # убрать колонку Category
```

Out[91]:

	Price	Rating	Rating2.0
0	0.0	4.1	-8.2
1	0.0	3.9	-7.8
2	0.0	4.7	-9.4
3	0.0	4.5	-9.0
4	0.0	4.3	-8.6
...	...	...	...
10834	0.0	4.0	-8.0
10836	0.0	4.5	-9.0
10837	0.0	5.0	-10.0
10839	0.0	4.5	-9.0
10840	0.0	4.5	-9.0

9366 rows × 3 columns

In [92]:
```
# target_df.drop("Rating2.0", axis=1, inplace=True)
target_df
```

Out[92]:

	Category	Price	Rating	Rating2.0
0	ART_AND_DESIGN	0.0	4.1	-8.2
1	ART_AND_DESIGN	0.0	3.9	-7.8
2	ART_AND_DESIGN	0.0	4.7	-9.4
3	ART_AND_DESIGN	0.0	4.5	-9.0
4	ART_AND_DESIGN	0.0	4.3	-8.6
...	...	...	...	...
10834	FAMILY	0.0	4.0	-8.0
10836	FAMILY	0.0	4.5	-9.0
10837	FAMILY	0.0	5.0	-10.0
10839	BOOKS_AND_REFERENCE	0.0	4.5	-9.0
10840	LIFESTYLE	0.0	4.5	-9.0

9366 rows × 4 columns

In [93]:
```
df.head()
```

Out[93]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10000	Free	0.0	Everyone
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500000	Free	0.0	Everyone
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5000000	Free	0.0	Everyone
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50000000	Free	0.0	Teen
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100000	Free	0.0	Everyone

In [94]:
```
1 target_df.head()
```

Out[94]:

	Category	Price	Rating	Rating2.0
0	ART_AND_DESIGN	0.0	4.1	-8.2
1	ART_AND_DESIGN	0.0	3.9	-7.8
2	ART_AND_DESIGN	0.0	4.7	-9.4
3	ART_AND_DESIGN	0.0	4.5	-9.0
4	ART_AND_DESIGN	0.0	4.3	-8.6

In [95]:
```
1 target_df.groupby(by="Category").mean()
```

Out[95]:

Category	Price	Rating	Rating2.0
ART_AND_DESIGN	0.096290	4.358065	-8.716129
AUTO_AND_VEHICLES	0.027260	4.190411	-8.380822
BEAUTY	0.000000	4.278571	-8.557143
BOOKS_AND_REFERENCE	0.134157	4.346067	-8.692135
BUSINESS	0.245512	4.121452	-8.242904
COMICS	0.000000	4.155172	-8.310345
COMMUNICATION	0.172835	4.158537	-8.317073
DATING	0.117744	3.970769	-7.941538
EDUCATION	0.115871	4.389032	-8.778065
ENTERTAINMENT	0.053557	4.126174	-8.252349
EVENTS	0.000000	4.435556	-8.871111
FAMILY	1.314677	4.192272	-8.384545
FINANCE	7.553777	4.131889	-8.263777
FOOD_AND_DRINK	0.077798	4.166972	-8.333945
GAME	0.255570	4.286326	-8.572653
HEALTH_AND_FITNESS	0.152795	4.277104	-8.554209
HOUSE_AND_HOME	0.000000	4.197368	-8.394737
LIBRARIES_AND_DEMO	0.000000	4.178462	-8.356923
LIFESTYLE	6.244841	4.094904	-8.189809
MAPS_AND_NAVIGATION	0.217339	4.051613	-8.103226
MEDICAL	2.980400	4.189143	-8.378286
NEWS_AND_MAGAZINES	0.017082	4.132189	-8.264378
PARENTING	0.191600	4.300000	-8.600000

PERSONALIZATION	0.406879	4.335987	-8.671975
PHOTOGRAPHY	0.278360	4.192114	-8.384227
PRODUCTIVITY	0.202051	4.211396	-8.422792
SHOPPING	0.023025	4.259664	-8.519328
SOCIAL	0.007645	4.255598	-8.511197
SPORTS	0.292194	4.223511	-8.447022
TOOLS	0.283243	4.047411	-8.094823
TRAVEL_AND_LOCAL	0.165885	4.109292	-8.218584
VIDEO_PLAYERS	0.065375	4.063750	-8.127500
WEATHER	0.392400	4.244000	-8.488000

In [96]:
```
target_df.groupby(by="Category").agg(["min", "median", "mean",
```

Out[96]:

	Price				Rating				R
	min	median	mean	max	min	median	mean	max	m
Category									
ART_AND_DESIGN	0.0	0.0	0.096290	1.99	3.2	4.4	4.358065	5.0	-1
AUTO_AND_VEHICLES	0.0	0.0	0.027260	1.99	2.1	4.3	4.190411	4.9	-
BEAUTY	0.0	0.0	0.000000	0.00	3.1	4.3	4.278571	4.9	-
BOOKS_AND_REFERENCE	0.0	0.0	0.134157	4.60	2.7	4.5	4.346067	5.0	-1
BUSINESS	0.0	0.0	0.245512	17.99	1.0	4.3	4.121452	5.0	-1
COMICS	0.0	0.0	0.000000	0.00	2.8	4.4	4.155172	5.0	-1
COMMUNICATION	0.0	0.0	0.172835	4.99	1.0	4.3	4.158537	5.0	-1
DATING	0.0	0.0	0.117744	7.99	1.0	4.1	3.970769	5.0	-1
EDUCATION	0.0	0.0	0.115871	5.99	3.5	4.4	4.389032	4.9	-
ENTERTAINMENT	0.0	0.0	0.053557	4.99	3.0	4.2	4.126174	4.7	-
EVENTS	0.0	0.0	0.000000	0.00	2.9	4.5	4.435556	5.0	-1
FAMILY	0.0	0.0	1.314677	399.99	1.0	4.3	4.192272	5.0	-1
FINANCE	0.0	0.0	7.553777	399.99	1.0	4.3	4.131889	5.0	-1
FOOD_AND_DRINK	0.0	0.0	0.077798	4.99	1.7	4.3	4.166972	5.0	-1
GAME	0.0	0.0	0.255570	17.99	1.0	4.4	4.286326	5.0	-1
HEALTH_AND_FITNESS	0.0	0.0	0.152795	7.99	1.4	4.5	4.277104	5.0	-1
HOUSE_AND_HOME	0.0	0.0	0.000000	0.00	2.8	4.3	4.197368	4.8	-
LIBRARIES_AND_DEMO	0.0	0.0	0.000000	0.00	3.1	4.2	4.178462	5.0	-1
LIFESTYLE	0.0	0.0	6.244841	400.00	1.5	4.2	4.094904	5.0	-1
MAPS_AND_NAVIGATION	0.0	0.0	0.217339	11.99	1.9	4.2	4.051613	4.9	-
MEDICAL	0.0	0.0	2.980400	79.99	1.0	4.3	4.189143	5.0	-1

NEWS_AND_MAGAZINES	0.0	0.0	0.017082	2.99	1.7	4.2	4.132189	5.0	-1
PARENTING	0.0	0.0	0.191600	4.99	2.0	4.4	4.300000	5.0	-1
PERSONALIZATION	0.0	0.0	0.406879	9.99	2.5	4.4	4.335987	5.0	-1
PHOTOGRAPHY	0.0	0.0	0.278360	19.99	2.0	4.3	4.192114	5.0	-1
PRODUCTIVITY	0.0	0.0	0.202051	8.99	1.0	4.3	4.211396	5.0	-1
SHOPPING	0.0	0.0	0.023025	2.99	1.6	4.3	4.259664	5.0	-1
SOCIAL	0.0	0.0	0.007645	0.99	1.9	4.3	4.255598	5.0	-1
SPORTS	0.0	0.0	0.292194	29.99	1.5	4.3	4.223511	5.0	-1
TOOLS	0.0	0.0	0.283243	14.99	1.0	4.2	4.047411	5.0	-1
TRAVEL_AND_LOCAL	0.0	0.0	0.165885	8.99	2.2	4.3	4.109292	5.0	-1
VIDEO_PLAYERS	0.0	0.0	0.065375	5.99	1.8	4.2	4.063750	4.9	-
WEATHER	0.0	0.0	0.392400	6.99	3.3	4.3	4.244000	4.8	-

In [97]:
```
1 target_df.groupby(by="Category").agg({"Price": "max", "Rating":
```

Out[97]:

Category	Price	Rating
ART_AND_DESIGN	1.99	4.358065
AUTO_AND_VEHICLES	1.99	4.190411
BEAUTY	0.00	4.278571
BOOKS_AND_REFERENCE	4.60	4.346067
BUSINESS	17.99	4.121452
COMICS	0.00	4.155172
COMMUNICATION	4.99	4.158537
DATING	7.99	3.970769
EDUCATION	5.99	4.389032
ENTERTAINMENT	4.99	4.126174
EVENTS	0.00	4.435556
FAMILY	399.99	4.192272
FINANCE	399.99	4.131889
FOOD_AND_DRINK	4.99	4.166972
GAME	17.99	4.286326
HEALTH_AND_FITNESS	7.99	4.277104
HOUSE_AND_HOME	0.00	4.197368
LIBRARIES_AND_DEMO	0.00	4.178462
LIFESTYLE	400.00	4.094904

MAPS_AND_NAVIGATION	11.99	4.051613
MEDICAL	79.99	4.189143
NEWS_AND_MAGAZINES	2.99	4.132189
PARENTING	4.99	4.300000
PERSONALIZATION	9.99	4.335987
PHOTOGRAPHY	19.99	4.192114
PRODUCTIVITY	8.99	4.211396
SHOPPING	2.99	4.259664
SOCIAL	0.99	4.255598
SPORTS	29.99	4.223511
TOOLS	14.99	4.047411
TRAVEL_AND_LOCAL	8.99	4.109292
VIDEO_PLAYERS	5.99	4.063750
WEATHER	6.99	4.244000

In [98]:
```
target_df.sort_values(by="Rating", ascending=False)
```

Out[98]:

	Category	Price	Rating	Rating2.0
9056	GAME	1.99	5.0	-10.0
8395	NEWS_AND_MAGAZINES	0.00	5.0	-10.0
8493	FAMILY	0.00	5.0	-10.0
6330	FAMILY	0.00	5.0	-10.0
6342	BUSINESS	0.00	5.0	-10.0
...	...	...	...	...
7806	BUSINESS	0.00	1.0	-2.0
10591	TOOLS	0.00	1.0	-2.0
7427	COMMUNICATION	0.00	1.0	-2.0
7926	FINANCE	0.00	1.0	-2.0
4127	FAMILY	2.99	1.0	-2.0

9366 rows × 4 columns

In [99]:
```
# можно отсортировать
target_df.groupby(by="Category", as_index=False,).max().sort_va
    by="Price", ascending=False
)
```

Out[99]:

	Category	Price	Rating	Rating2.0
18	LIFESTYLE	400.00	5.0	-3.0

12	FINANCE	399.99	5.0	-2.0
11	FAMILY	399.99	5.0	-2.0
20	MEDICAL	79.99	5.0	-2.0
28	SPORTS	29.99	5.0	-3.0
24	PHOTOGRAPHY	19.99	5.0	-4.0
4	BUSINESS	17.99	5.0	-2.0
14	GAME	17.99	5.0	-2.0
29	TOOLS	14.99	5.0	-2.0
19	MAPS_AND_NAVIGATION	11.99	4.9	-3.8
23	PERSONALIZATION	9.99	5.0	-5.0
30	TRAVEL_AND_LOCAL	8.99	5.0	-4.4
25	PRODUCTIVITY	8.99	5.0	-2.0
15	HEALTH_AND_FITNESS	7.99	5.0	-2.8
7	DATING	7.99	5.0	-2.0
32	WEATHER	6.99	4.8	-6.6
31	VIDEO_PLAYERS	5.99	4.9	-3.6
8	EDUCATION	5.99	4.9	-7.0
13	FOOD_AND_DRINK	4.99	5.0	-3.4
9	ENTERTAINMENT	4.99	4.7	-6.0
22	PARENTING	4.99	5.0	-4.0
6	COMMUNICATION	4.99	5.0	-2.0
3	BOOKS_AND_REFERENCE	4.60	5.0	-5.4
26	SHOPPING	2.99	5.0	-3.2
21	NEWS_AND_MAGAZINES	2.99	5.0	-3.4
0	ART_AND_DESIGN	1.99	5.0	-6.4
1	AUTO_AND_VEHICLES	1.99	4.9	-4.2
27	SOCIAL	0.99	5.0	-3.8
17	LIBRARIES_AND_DEMO	0.00	5.0	-6.2
10	EVENTS	0.00	5.0	-5.8
5	COMICS	0.00	5.0	-5.6
2	BEAUTY	0.00	4.9	-6.2
16	HOUSE_AND_HOME	0.00	4.8	-5.6

In [100]:
```
1 df.head()
```

Out[100]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10000	Free	0.0	Everyone
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500000	Free	0.0	Everyone
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5000000	Free	0.0	Everyone
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50000000	Free	0.0	Teen
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100000	Free	0.0	Everyone

In [101]:
```
target_df_2 = df[["Category", "Price", "Rating", "Content Ratin
target_df_2.dropna(inplace=True)
target_df_2
```

```
<ipython-input-101-8dd32e3e01ed>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy)
  target_df_2.dropna(inplace=True)
```

Out[101]:

	Category	Price	Rating	Content Rating
0	ART_AND_DESIGN	0.0	4.1	Everyone
1	ART_AND_DESIGN	0.0	3.9	Everyone
2	ART_AND_DESIGN	0.0	4.7	Everyone
3	ART_AND_DESIGN	0.0	4.5	Teen
4	ART_AND_DESIGN	0.0	4.3	Everyone
...	...	...	...	...
10834	FAMILY	0.0	4.0	Everyone
10836	FAMILY	0.0	4.5	Everyone
10837	FAMILY	0.0	5.0	Everyone
10839	BOOKS_AND_REFERENCE	0.0	4.5	Mature 17+
10840	LIFESTYLE	0.0	4.5	Everyone

9366 rows × 4 columns

In [102]:
```
target_df_2["Content Rating"].value_counts()
```

Out[102]:
```
Everyone           7420
Teen               1084
Mature 17+          461
Everyone 10+        397
Adults only 18+       3
Unrated               1
Name: Content Rating, dtype: int64
```

In [103]:
```
target_df_2.groupby(["Category", "Content Rating"]).mean()
```

Out[103]:

Category	Content Rating	Price	Rating
ART_AND_DESIGN	**Everyone**	0.102931	4.346552
	Everyone 10+	0.000000	4.700000
	Teen	0.000000	4.466667
AUTO_AND_VEHICLES	**Everyone**	0.028028	4.188732
	Everyone 10+	0.000000	4.300000
...	**...**	...	...
VIDEO_PLAYERS	**Teen**	0.000000	4.087500
WEATHER	**Everyone**	0.414507	4.229577
	Everyone 10+	0.000000	4.400000
	Mature 17+	0.000000	4.700000
	Teen	0.000000	4.450000

113 rows × 2 columns

In [104]:
```
target_df_2.groupby(["Category", "Content Rating"], as_index=Fa
```

Out[104]:

	Category	Content Rating	Price	Rating
0	ART_AND_DESIGN	Everyone	0.102931	4.346552
1	ART_AND_DESIGN	Everyone 10+	0.000000	4.700000
2	ART_AND_DESIGN	Teen	0.000000	4.466667
3	AUTO_AND_VEHICLES	Everyone	0.028028	4.188732
4	AUTO_AND_VEHICLES	Everyone 10+	0.000000	4.300000
...	...	...	...	...
108	VIDEO_PLAYERS	Teen	0.000000	4.087500
109	WEATHER	Everyone	0.414507	4.229577
110	WEATHER	Everyone 10+	0.000000	4.400000
111	WEATHER	Mature 17+	0.000000	4.700000
112	WEATHER	Teen	0.000000	4.450000

113 rows × 4 columns

In [105]:
```
target_df_2.to_csv("Category_Content.csv")  # сохраним наш резу
```

Сводная таблица

In [106]:

```
pd.pivot_table(
    target_df_2,
    values="Rating",
    index="Category",
    columns="Content Rating",
    aggfunc=np.mean,
).fillna(0)
```

Out[106]:

Content Rating	Adults only 18+	Everyone	Everyone 10+	Mature 17+	Teen	Unrated
Category						
ART_AND_DESIGN	0.0	4.346552	4.700000	0.000000	4.466667	0.0
AUTO_AND_VEHICLES	0.0	4.188732	4.300000	0.000000	4.200000	0.0
BEAUTY	0.0	4.287179	0.000000	4.500000	4.000000	0.0
BOOKS_AND_REFERENCE	0.0	4.351333	4.460000	4.166667	4.305000	0.0
BUSINESS	0.0	4.119064	0.000000	0.000000	4.300000	0.0
COMICS	4.2	4.344000	4.450000	3.771429	4.031818	0.0
COMMUNICATION	0.0	4.140678	0.000000	4.311111	4.320833	0.0
DATING	0.0	4.100000	0.000000	3.975978	3.600000	0.0
EDUCATION	0.0	4.385315	4.487500	4.166667	4.800000	0.0
ENTERTAINMENT	0.0	4.176923	4.100000	4.200000	4.100000	0.0
EVENTS	0.0	4.411111	4.500000	0.000000	4.542857	0.0
FAMILY	0.0	4.194157	4.238889	4.193182	4.157438	0.0
FINANCE	0.0	4.132602	0.000000	0.000000	4.075000	0.0
FOOD_AND_DRINK	0.0	4.147000	4.300000	0.000000	4.414286	0.0
GAME	0.0	4.283480	4.346923	4.262162	4.272531	0.0
HEALTH_AND_FITNESS	0.0	4.280899	3.983333	4.385714	4.276471	0.0
HOUSE_AND_HOME	0.0	4.185135	0.000000	0.000000	4.650000	0.0
LIBRARIES_AND_DEMO	0.0	4.178462	0.000000	0.000000	0.000000	0.0
LIFESTYLE	0.0	4.073759	3.840000	4.255556	4.416667	0.0
MAPS_AND_NAVIGATION	0.0	4.057025	0.000000	2.700000	4.400000	0.0
MEDICAL	0.0	4.176276	4.387500	4.466667	4.533333	0.0
NEWS_AND_MAGAZINES	0.0	4.120000	4.079688	4.307692	4.216129	0.0
PARENTING	0.0	4.285417	0.000000	4.600000	4.700000	0.0
PERSONALIZATION	0.0	4.323443	4.380000	4.500000	4.403571	0.0
PHOTOGRAPHY	0.0	4.185000	0.000000	4.380000	4.291667	0.0
PRODUCTIVITY	0.0	4.212968	3.600000	4.600000	4.050000	0.0
SHOPPING	0.0	4.239037	0.000000	4.266667	4.339583	0.0
SOCIAL	0.0	4.304706	4.100000	4.122727	4.301887	0.0

SPORTS	4.5	4.217091	4.333333	4.285714	4.140000	0.0
TOOLS	0.0	4.044704	0.000000	3.700000	4.500000	4.1
TRAVEL_AND_LOCAL	0.0	4.108182	0.000000	4.600000	4.060000	0.0
VIDEO_PLAYERS	0.0	4.064394	4.150000	3.650000	4.087500	0.0
WEATHER	0.0	4.229577	4.400000	4.700000	4.450000	0.0

Работа с БД

In [107]:
```
# !pip3 install psycopg2-binary
# https://www.psycopg.org/docs/install.html#quick-install
```

In [108]:
```
import psycopg2
```

In [109]:

```
conn = psycopg2.connect(
    host="158.160.52.106",
    port=5432,
    database="postgres",
    user="student",
    password="JvLda93aA",
)
cur = conn.cursor()

cur.execute("SELECT * FROM msu_analytics.game")


data = cur.fetchall()
data[:5]
```

Out[109]:

```
[(15,
  13,
  11,
  datetime.datetime(2022, 11, 17, 2, 13, 10, 970497),
  Decimal('2701.92'),
  0,
  0,
  datetime.time(0, 0)),
 (79,
  31,
  3,
  datetime.datetime(2022, 12, 2, 19, 53, 58, 107147),
  Decimal('1399.31'),
  0,
  0,
  datetime.time(0, 0)),
 (111,
  21,
  14,
  datetime.datetime(2022, 11, 22, 9, 31, 30, 506502),
  Decimal('1349.32'),
  0,
  0,
  datetime.time(0, 0)),
 (127,
  15,
  7,
  datetime.datetime(2022, 12, 27, 1, 59, 28, 540922),
  Decimal('1557.79'),
  0,
  0,
  datetime.time(0, 0)),
 (143,
  22,
  8,
  datetime.datetime(2022, 12, 30, 10, 39, 51, 834731),
  Decimal('1880.07'),
  0,
  0,
  datetime.time(0, 0))]
```

In [110]:
```
1 pd.read_sql("SELECT * FROM msu_analytics.game", conn).head()
```

Out[110]:

	game_rk	quest_rk	employee_rk	game_dttm	price	game_flg	finish_flg	time
0	15	13	11	2022-11-17 02:13:10.970497	2701.92	0	0	00:00:00
1	79	31	3	2022-12-02 19:53:58.107147	1399.31	0	0	00:00:00
2	111	21	14	2022-11-22 09:31:30.506502	1349.32	0	0	00:00:00
3	127	15	7	2022-12-27 01:59:28.540922	1557.79	0	0	00:00:00
4	143	22	8	2022-12-30 10:39:51.834731	1880.07	0	0	00:00:00

Регулярные выражения

Регулярное выражение -- правило / шаблон поиска подстроки в тексте

Шаблон	Описание	Пример	Применяем к тексту
`.`	Один любой символ, кроме новой строки `\n`.	`м.л.ко`	молоко, малако, Им0л0коИхлеб
`\d`	Любая цифра	`СУ\d\d`	СУ35, СУ111, АЛСУ14
`\D`	Любой символ, кроме цифры	`926\D123`	926)123, 1926-1234
`\s`	Любой пробельный символ (пробел, табуляция, конец строки и т.п.)	`бор\soda`	бор ода, бор ода, борода
`\S`	Любой непробельный символ	`\S123`	X123, я123, !123456, 1 + 123456
`\w`	Любая буква (то, что может быть частью слова), а также цифры и _	`\w\w\w`	Год, f_3, qwert
`\W`	Любая не-буква, не-цифра и не подчёркивание	`сом\W`	сом!, сом?
`[..]`	Один из символов в скобках, а также любой символ из диапазона `a-b`	`[0-9][0-9A-Fa-f]`	12, 1F, 4B
`[^..]`	Любой символ, кроме перечисленных	`<[^>]>`	<1>, <a>, <>
`\d≈[0-9]`, `\D≈[^0-9]`, `\w≈[0-9a-zA-Za-яА-ЯёЁ]`, `\s≈[\f\n\r\t\v]`	Буква “ё” не включается в общий диапазон букв! Вообще говоря, в `\d` включается всё, что в юникое помечено как «цифра», а в `\w` — как буква. Ещё много всего!		
`[abc-]`, `[-1]`	если нужен минус, его нужно указать последним или первым		
`[*[(+\\\]\t]`	внутри скобок нужно экранировать только] и \		

Ссылка: https://medium.com/@enduranceprog/regular-expression-2b074b4bc68a (https://medium.com/@enduranceprog/regular-expression-2b074b4bc68a)

Больше по синтаксису: https://docs.python.org/3/library/re.html#regular-expression-syntax (https://docs.python.org/3/library/re.html#regular-expression-syntax)

Где можно встретить:

- Валидация данных (соответствие пароля, email'а шаблону)
- Сбор данных (веб-скрапинг)
- Парсинг и обработка данных
- Простая замена строк
- ...

In [111]:
```
import re  # пакет для работы с регулярными выражениями
```

In [112]:
```
text = """Центральный банк КНР \\ увеличил норму резервирования
Форвардный контракт — это контракт (фьючерс), фиксирующий обяза
Это финансовый инструмент, позволяющий вам подстраховаться на с
Например, текущий базовый курс доллара 59 Р за доллар, а по фор
```

In [113]:
```
# вернет первое найденное соответствие Центральный, с начала ст
re.search("^Центральный", text)
```

Out[113]: <re.Match object; span=(0, 11), match='Центральный'>

In [114]:
```
re.search("^\w+", text)  # первое слово
```

Out[114]: <re.Match object; span=(0, 11), match='Центральный'>

In [115]:
```
re.match("^\w+", text)  # match всегда работает для начала
```

Out[115]: <re.Match object; span=(0, 11), match='Центральный'>

In [116]:
```
re.search("70 Р.$", text)  # $ строка оканчивается на 70 Р.
```

Out[116]: <re.Match object; span=(743, 748), match='70 Р.'>

In [117]:
```
re.search("..\.$", text)  # на самом деле . это спец символ
```

Out[117]: <re.Match object; span=(745, 748), match=' Р.'>

In [118]:
```
re.search("\\\\", text)  # re.search("\\", text)
```

Out[118]: <re.Match object; span=(21, 22), match='\\'>

In [119]:
```
re.findall(r"\w", text[:20])  # найдем все шаблоны, посмотрим н
```

Out[119]:
```
['Ц',
 'е',
 'н',
 'т',
 'р',
 'а',
 'л',
 'ь',
 'н',
 'ы',
 'й',
 'б',
 'а',
 'н',
 'к',
 'К',
 'Н',
 'Р']
```

In [120]:
```
# {} -- точное количество
print(
    re.findall(r"\w{4}", text[:20]),
    re.findall(r"\w{2,4}", text[:20]),
    re.findall(r"\w{5,}", text[:20]),
)
```

```
['Цент', 'раль', 'банк'] ['Цент', 'раль', 'ный', 'банк', 'КНР'] ['
Центральный']
```

In [121]:
```
re.findall(r"\w+", text[:30])  # все слова
```

Out[121]:
```
['Центральный', 'банк', 'КНР', 'увеличи']
```

In [122]:
```
re.findall(r"[бв][а-я]+", text)  # только слова без цифр
```

Out[122]:
```
['банк',
 'величил',
 'вирования',
 'вардным',
 'вардный',
 'бязательства',
 'бретателя',
 'валюту',
 'ванному',
 'время',
 'виях',
 'вый',
 'воляющий',
 'вам',
 'ваться',
 'баний',
 'валют',
 'вать',
 'вардный',
 'ва',
 'будущем',
 'базовый',
 'вардному',
 'вперед',
 'вляет',
 'бно',
 'бы',
 'будет',
 'выше']
```

In [123]:
```
re.findall(r"\d+", text)  # только цифры
```

Out[123]:
```
['20', '59', '3', '59', '5', '3', '59', '5', '70']
```

In [124]:
```
re.sub(r"[^\w\s]", "", text)  # замена знаков пунктуации
```

Out[124]:
```
'Центральный банк КНР  увеличил норму резервирования по форвардным
контрактам на курс доллара с нуля до 20\nФорвардный контракт  это
контракт фьючерс фиксирующий обязательства приобретателя этого кон
тракта купить или продать иностранную валюту по фиксированному кур
су в определенное время на определенных условиях\nЭто финансовый и
нструмент позволяющий вам подстраховаться на случай колебаний валю
т или захеджировать риски То есть форвардный контракт  это цена ак
тива в будущем\nНапример текущий базовый курс доллара 59 Р за долл
ар а по форвардному контракту на 3 месяца вперед курс составляет 5
95 Р Удобно иметь такой контракт чтобы купить доллар через 3 месяц
а по 595 Р если в этот момент курс будет уже сильно выше например
70 Р'
```

In [125]:
```
re.split(r"[;,]", "раз;два,три")  # разбиваем строку по несколь
```

Out[125]:
```
['раз', 'два', 'три']
```

In [126]:

```
words = re.compile(r"\w+")
nums = re.compile(r"\d+")

re.findall(words, re.sub(nums, "", text[:100]))
```

Out[126]:

```
['Центральный',
 'банк',
 'КНР',
 'увеличил',
 'норму',
 'резервирования',
 'по',
 'форвардным',
 'контрактам',
 'на',
 'курс',
 'доллара',
 'с',
 'нуля']
```